ARMY RESEARCH LABORATORY

Reading, Writing, and Analyzing ZDATA Files Using C++

by Robert J. Yager

ARL-TN-541 **October 2013**

Approved for public release; distribution is unlimited.

NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

Army Research Laboratory
Aberdeen Proving Ground, MD 21005-5066

ARL-TN-541 **October 2013**

Reading, Writing, and Analyzing ZDATA Files Using C++

Robert J. Yager
Weapons and Materials Research Directorate, ARL

Approved for public release; distribution is unlimited.

REPORT DOCUMENTATION PAGE	Form Approved OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing the burden, to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.
PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED (From - To)
October 2013	Final	June 2012–March 2013

4. TITLE AND SUBTITLE
Reading, Writing, and Analyzing ZDATA Files Using C++

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Robert J. Yager

5d. PROJECT NUMBER
AH80

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
U.S. Army Research Laboratory
ATTN: RDRL-WML-A
Aberdeen Proving Ground, MD 21005-5066

8. PERFORMING ORGANIZATION REPORT NUMBER
ARL-TN-541

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT
ZDATA files can be used to store information about the fragments produced by fragmentation weapons. This report presents a set of functions, written in C++, that can be used to read, write, and analyze ZDATA files.

15. SUBJECT TERMS
ZDATA, C++, analyze, read, write

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Robert J. Yager
Unclassified	Unclassified	Unclassified	UU	20	19b. TELEPHONE NUMBER (*Include area code*) 410-278-6689

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39.18

Contents

Acknowledgments

The author would like to thank Mary Arthur of the U.S. Army Research Laboratory's Weapons and Materials Research Directorate. Ms. Arthur provided technical and editorial recommendations that improved the quality of this report.

1. Introduction

ZDATA files can be used to store information about the fragments produced by fragmentation weapons. This report presents a set of functions, written in C++, that can be used to read, write, and analyze ZDATA files. Functions from the yIo namespace[1] are used in examples. A summary sheet is provided at the end of this report. It presents the yZdata namespace, which contains the two structs and three functions that are described in this report.

2. ZDATA File Format

With respect to ZDATA files, fragmentation weapons, and thus their fragment patterns, are assumed to be axially symmetric. Fragment characteristics are grouped by angular zones (see figure 1 inset). Within each angular zone, initial fragment speeds, as well as mass and count distributions, are documented. Figure 1 presents an annotated sample ZDATA file that is in standard format.[2]

[1]Yager, R. J. *Reading, Writing, and Parsing Text Files Using C++*; ARL-TN-545; U.S. Army Research Laboratory: Aberdeen Proving Ground, MD, June 2013.

[2]Joint Technical Coordinating Group for Munitions Effectiveness. *Computer Program for General Full Spray Personnel MAE Computations: Volume 1 – Users Manual*; 61 JTCG/ME-79-6-1; U.S. Army Materiel Systems Analysis Activity: Aberdeen Proving Ground, MD, April 1991.

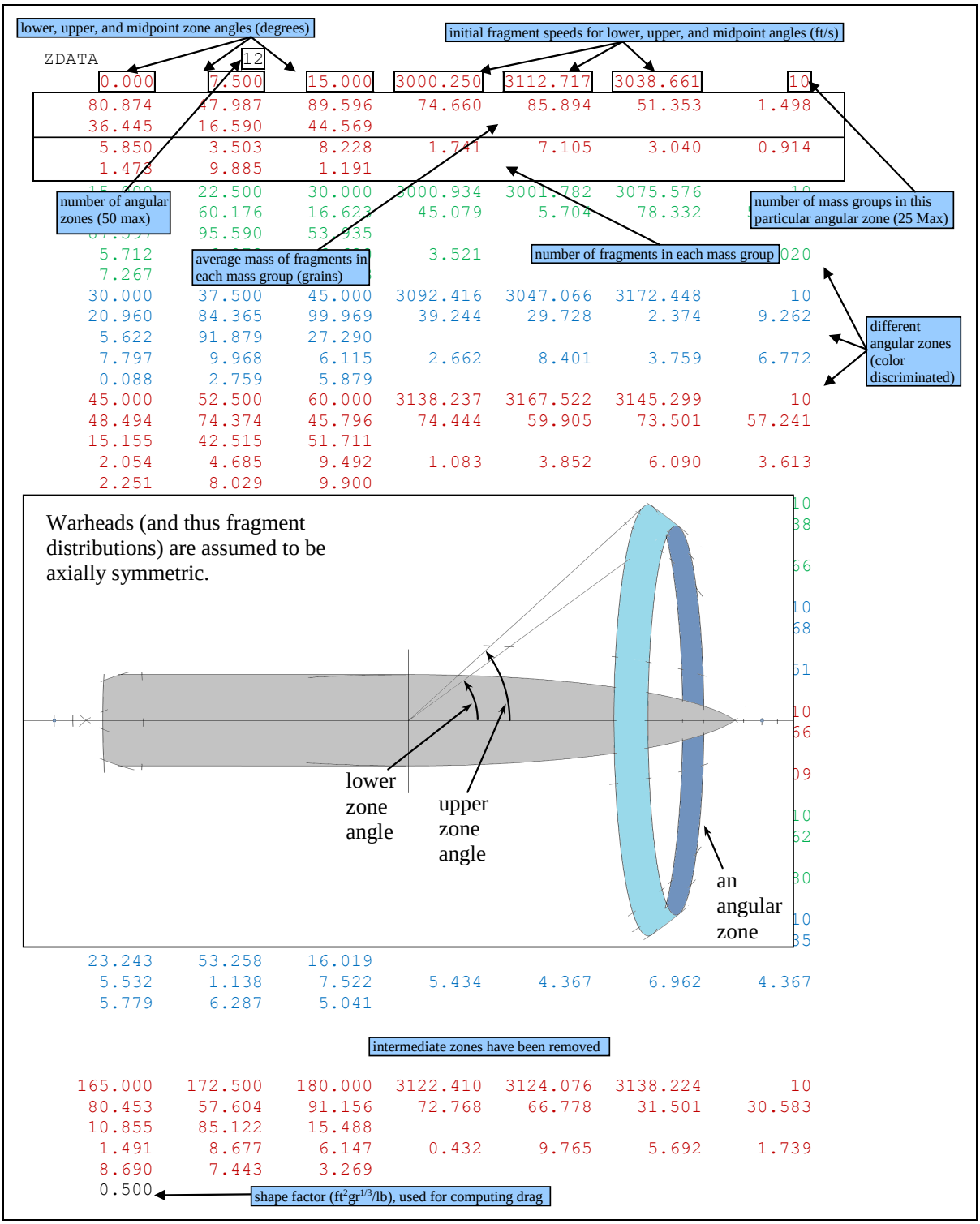

Figure 1. Annotated sample ZDATA file with inset visual of an angular zone.

3. ZDATA Structs

ZDATA structs can be used to store the information contained in ZDATA files. They can be created and populated either directly or through the use of the StringToZdata() function (see section 5.1 for more information).

The user is free to store the values of a ZDATA struct in whatever units he/she chooses. If a ZDATA struct is created using the StringToZdata() function, the units from the source character array will be retained.

ZDATA-Struct Code

```
struct ZDATA{//<=====STORES THE INFORMATION CONTAINED IN A STANDARD ZDATA FILE
  std::vector<double>AL;//<------------------------LOWER ANGLES [ANGLE INDEX]
  std::vector<double>AM;//<----------------------MIDPOINT ANGLES [ANGLE INDEX]
  std::vector<double>AU;//<------------------------UPPER ANGLES [ANGLE INDEX]
  std::vector<double>VL;//<--------------SPEEDS FOR LOWER ANGLES [ANGLE INDEX]
  std::vector<double>VM;//<-----------SPEEDS FOR MIDPOINT ANGLES [ANGLE INDEX]
  std::vector<double>VU;//<--------------SPEEDS FOR UPPER ANGLES [ANGLE INDEX]
  std::vector<std::vector<double> >M;//--MASS VALUES [ANGLE INDEX][MASS INDEX]
  std::vector<std::vector<double> >N;//<------COUNTS [ANGLE INDEX][MASS INDEX]
  double sf;//<-------------------------SHAPE FACTOR (USED FOR COMPUTING DRAG)
};//~~~YAGENAUT@GMAIL.COM~~~~~~~~~~~~~~~~~~~~~~~~~LAST~UPDATED~16SEP2013~~~~~~
```

ZDATA-Struct Members

AL — **AL** stores all of the lower-angle values for a ZDATA file (one per angular bin).

AM — **AM** stores all of the midpoint-angle values for a ZDATA file (one per angular bin).

AU — **AU** stores all of the upper-angle values for a ZDATA file (one per angular bin).

VL — **VL** stores all of the speeds that are associated with the lower-angle values.

VM — **VM** stores all of the speeds that are associated with the midpoint-angle values.

VU — **VU** stores all of the speeds that are associated with the upper-angle values.

M	**M** stores all of the fragment-mass values from a ZDATA file, indexed first by angular bin, then by mass bin. The number of mass bins per angular bin can vary between angular bins.
N	**N** stores all of the fragment-count values from a ZDATA file, indexed first by angular bin, then by mass bin. The number of mass bins per angular bin can vary between angular bins. Since the fragment-count values in a ZDATA file represent expected values, they are not required to be integer values.
sf	**sf** stores the shape factor for a ZDATA file.

4. Creating ZDATA Files

Creating ZDATA files using the yZdata namespace is a two-step process. First, the ZdataToString() function is used to create a character array from information that is stored in a ZDATA struct. Once the character array has been created, it can be written to a file in a variety of ways. The example included at the end of this section uses the WriteTextFile() function from the yIo namespace.

4.1 ZdataToString() Function

The ZdataToString() function creates a character array that contains the information contained in a ZDATA struct. Character arrays that are created using the ZdataToString() function retain the units from the source ZDATA struct.

Note that the ZdataToString() function uses the “new” command to allocate memory for the character array that is pointed to by the return value. Thus, to avoid memory leaks, each use of the ZdataToString() function should be accompanied by a use of the “delete[]” operator.

ZdataToString() Code

```
inline char*ZdataToString(//<==========CREATES A CHAR ARRAY FROM A ZDATA STRUCT
    const ZDATA&Z,//<-----------------------------------------A ZDATA STRUCT
    int d=3){//<-THE NUMBER OF DIGITS AFTER THE DECIMAL POINT FOR NON INTEGERS
  int m=Z.M.size(),l=32;/*<-*/for(int i=0;i<m;++i)l+=73+142*Z.M[i].size()/7;
  char*s=new char[l+1];
  s+=sprintf(s,"ZDATA     %10d\n",m);//.............................header line
  for(int i=0,n=Z.M[0].size();i<m;++i,n=Z.M[i].size()){
    s+=sprintf(s," %9.*f %9.*f %9.*f %9.*f %9.*f %9.*f%10d\n",d,Z.AL[i],d,
      Z.AM[i],d,Z.AU[i],d,Z.VL[i],d,Z.VM[i],d,Z.VU[i],Z.M[i].size());
    for(int j=0;j<n;++j)
      s+=sprintf(s," %9.*f%s",d,Z.M[i][j],j%7==6||j==n-1?"\n":"");//....masses
    for(int j=0;j<n;++j)
      s+=sprintf(s," %9.*f%s",d,Z.N[i][j],j%7==6||j==n-1?"\n":"");}//...counts
  s+=sprintf(s," %9.*f\n",d,Z.sf);//...............................shape factor
  return s-l;//...................note that s points to newly allocated memory
}//~~~~YAGENAUT@GMAIL.COM~~~~~~~~~~~~~~~~~~~~~~~~~LAST~UPDATED~16SEP2013~~~~~~
```

ZdataToString() Parameters

Z — **Z** specifies a ZDATA struct.

d — **d** specifies the number of digits to the right of the decimal for non-integer values in the output character array. The default value is 3.

ZdataToString() Return Value

The ZdataToString() function returns a pointer to a character array that contains the information from the input ZDATA struct.

4.2 Example: Creating a Sample ZDATA File

The following example was used to create the sample file “zdata.txt,” which is presented in figure 1. The example begins by creating a ZDATA struct that contains randomly drawn masses, counts, and speeds. The example then uses the ZdataToString() function to create a character array that contains the information from the newly created ZDATA struct. The WriteTextFile() function, borrowed from the yIo namespace, is then used to create the file “zdata.txt.” Finally, the “delete[]” operator is used to deallocate the memory that was allocated by the ZdataToString() function.

```
#include <cstdlib>//.....rand(),RAND_MAX
#include "y_zdata.h"//.....yZdata,<vector>
#include "y_io.h"//.....yIo
int main(){
  using std::vector;
  int m=12,n=10;//number of angular bins and number of mass bins per angular bin
  yZdata::ZDATA Z={vector<double>(m),vector<double>(m),vector<double>(m),//setup
    vector<double>(m),vector<double>(m),vector<double>(m),//                a
    vector<vector<double> >(m,vector<double>(n)),//                       ZDATA
    vector<vector<double> >(m,vector<double>(n)),.5};//                   struct
  for(int i=0;i<m;++i){//.....populate the ZDATA struct
    Z.AL[i]=i*180./m,Z.AM[i]=(i+.5)*180./m,Z.AU[i]=(i+1)*180./m;
    Z.VL[i]=3000+rand()*200./RAND_MAX;//.....set upper, middle, & lower
    Z.VM[i]=3000+rand()*200./RAND_MAX;//                speeds to random values
    Z.VU[i]=3000+rand()*200./RAND_MAX;//                  between 3000 and 3200
    for(int j=0;j<n;++j){
      Z.M[i][j]=rand()*100./RAND_MAX;//.....set masses to random values 0 to 100
      Z.N[i][j]=rand()*10./RAND_MAX;}}//.....set counts to random values 0 to 10
  char* s=yZdata::ZdataToString(Z,3);
  yIo::WriteTextFile("zdata.txt",s);
  delete[]s;
}//~~~~~~YAGENAUT@GMAIL.COM~~~~~~~~~~~~~~~~~~~~~~~~~LAST~UPDATED~16SEP2013~~~~~~
```

5. Reading ZDATA Files

ZDATA files can be read and interpreted by first reading a file into a character array and then using the StringToZdata() function to create a ZDATA struct.

5.1 StringToZdata() Function

The StringToZdata() function uses a character array that contains text formatted as shown in figure 1 to create a ZDATA struct. The input character array is modified in the process. ZDATA structs that are created using the StringToZdata() function retain the units from the source ZDATA character array.

The formatting standard for ZDATA files allows for cases where there is no white space between numbers (the numbers are column aligned). However, the StringToZdata() function can only correctly read files that have white space between numbers. Thus, if the user wishes to use the StringToZdata() function to read a ZDATA character array that does not have white space between numbers, the user will be responsible for inserting white spaces.

StringToZdata() Code

```
  inline ZDATA StringToZdata(//<======CREATES A ZDATA STRUCT FROM A ZDATA STRING
      char*s){//<--------------------A POINTER TO A STRING READ FROM A ZDATA FILE
    char d[6]="\n\r\f \t";//.............delimiting characters used in ZDATA file
    strtok(s,d);
    int m=atoi(strtok(NULL,d));//.........................number of angular bins
    ZDATA Z={std::vector<double>(m),std::vector<double>(m),
      std::vector<double>(m),std::vector<double>(m),std::vector<double>(m),
      std::vector<double>(m),std::vector<std::vector<double> >(m),
      std::vector<std::vector<double> >(m)};
    for(int i=0;i<m;++i){
      Z.AL[i]=atof(strtok(NULL,d)),Z.AM[i]=atof(strtok(NULL,d)),
      Z.AU[i]=atof(strtok(NULL,d)),Z.VL[i]=atof(strtok(NULL,d)),
      Z.VM[i]=atof(strtok(NULL,d)),Z.VU[i]=atof(strtok(NULL,d));
      int n=atoi(strtok(NULL,d));//...number of mass bins in the jth angular bin
      for(int j=0;j<n;++j)Z.M[i].push_back(atof(strtok(NULL,d)));//.......masses
      for(int j=0;j<n;++j)Z.N[i].push_back(atof(strtok(NULL,d)));}//......counts
    Z.sf=atof(strtok(NULL,d));//.....................................shape factor
    return Z;
  }//~~~~YAGENAUT@GMAIL.COM~~~~~~~~~~~~~~~~~~~~~~~~~~~LAST~UPDATED~16SEP2013~~~~~~
```

StringToZdata() Parameters

s — **s** points to a character array that contains the text that will be used to create a ZDATA struct.

StringToZdata() Return Value

The StringToZdata() function returns a ZDATA struct.

5.2 Example: Reading a Sample ZDATA File

The following example first uses the ReadTextFile() function from the yIo namespace to read the file "zdata.txt," presented in figure 1, into a character array. The character array is then used to create a ZDATA struct using the StringToZdata() function. Finally, the first five counts from the first angular bin are printed to the screen.

```
#include "y_zdata.h"//.............................yZdata,<cstdio>{printf()}
#include "y_io.h"//.................................................yIo
int main(){
  char*s=yIo::ReadTextFile("zdata.txt");
  yZdata::ZDATA Z=yZdata::StringToZdata(s);
  for(int j=0;j<4;++j)printf("%f , ",Z.N[0][j]);
  printf("%f\n",Z.N[0][4]);
  delete[]s;
}//~~~~~~YAGENAUT@GMAIL.COM~~~~~~~~~~~~~~~~~~~~~~~~~~~LAST~UPDATED~16SEP2013~~~~~~
```

OUTPUT:

```
8.087400 , 4.798700 , 8.959600 , 7.466000 , 8.589400
```

6. ZSTATS Structs

ZSTATS structs can be used to hold a set of variables that summarizes the contents of a ZDATA file. The AnalyzeZdata() function can be used to create a ZSTATS struct (see section 7.1 for more information). Note that the user is free to store the values of a ZSTATS struct in whatever units he/she chooses. If a ZSTATS struct is created using the AnalyzeZdata() function, then the units from the source ZDATA struct will be retained.

ZSTATS Struct Code

```
struct ZSTATS{//<==============A SET OF VARIABLES THAT SUMMARIZES A ZDATA FILE
  double M;//<----------------------------------------TOTAL MASS OF FRAGMENTS
  double N;//<---------------------------------------TOTAL NUMBER OF FRAGMENTS
  double KE;//<---------------------------------------TOTAL FRAGMENT ENERGY
  double m_avg;//<-----------------------------------AVERAGE MASS OF FRAGMENTS
  double v_avg;//<-----------------------------------AVERAGE FRAGMENT SPEED
};//~~~YAGENAUT@GMAIL.COM~~~~~~~~~~~~~~~~~~~~~~~~~LAST~UPDATED~16SEP2013~~~~~~
```

ZSTATS Struct Members

M — **M** is the total mass of all of the fragments that are specified in a ZDATA file. M is calculated by summing the product of count and mass values for all mass bins and all angular bins.

N — **N** is the total number of fragments that are specified in a ZDATA file. Since the counts in a ZDATA file represent expected values, **N** is not required to be an integer value. **N** is calculated by summing all counts for all mass bins and all angular bins.

KE — **KE** is the total fragment kinetic energy for an entire ZDATA file. For a given angular bin, the average of the squares of the upper, midpoint, and lower speeds is used to calculate **KE**.

m_avg — **m_avg** is the average fragment mass for an entire ZDATA file. **m_avg** is calculated by dividing **M** by **N**.

v_avg — **v_avg** represents a weighted average fragment speed for an entire ZDATA file. **v_avg** is calculated using the equation $\mathbf{KE}=.5\cdot\mathbf{M}\cdot\mathbf{v\_avg}^2$.

7. Analyzing ZDATA Files

7.1 AnalyzeZdata() Function

The AnalyzeZdata() function can be used to get an overview of the general characteristics of the fragments that are specified in ZDATA structs. The parameters that are calculated using the AnalyzeZdataFile() function are far from exhaustive; rather, they are meant to serve as a core set of descriptive values. Users may desire additional parameters, such as standard deviations, minimum and maximum values, median values, etc. Although the AnalyzeZdata() function does not calculate those parameters, its code can be used as a template for writing custom functions that further analyze the data found in ZDATA structs.

AnalyzeZdata() Code

```
inline ZSTATS AnalyzeZdata(//<======CREATE A ZSTATS STRUCT FROM A ZDATA STRUCT
    const ZDATA&Z){//<---------------------------CREATE USING ReadZdataFile()
  ZSTATS S={0,0,0,0,0};//.....................................initialize values
  for(int i=0,m=Z.M.size();i<m;++i){//..............loop through angular zones
    double v_avg_2=(Z.VL[i]*Z.VL[i]+Z.VM[i]*Z.VM[i]+Z.VU[i]*Z.VU[i])/3;
    for(int j=0,n=Z.M[i].size();j<n;++j){//.............loop through mass bins
      S.N+=Z.N[i][j];//.................................total fragment count
      S.M+=Z.N[i][j]*Z.M[i][j];//.........................total fragment mass
      S.KE+=.5*Z.M[i][j]*Z.N[i][j]*v_avg_2;}}//...........total kinetic energy
  S.m_avg=S.M/S.N;//.................................average fragment mass
  S.v_avg=sqrt(2*S.KE/S.M);//.........................average fragment speed
  return S;
}//~~~~YAGENAUT@GMAIL.COM~~~~~~~~~~~~~~~~~~~~~~~~~~LAST~UPDATED~16SEP2013~~~~~~
```

AnalyzeZdata() Parameters

Z **Z** specifies a ZDATA struct.

AnalyzeZdata() Return Value

The AnalyzeZdata() function returns a ZSTATS struct.

7.2 Example: Analyzing a Sample ZDATA File

The following example first uses the ReadTextFile() function from the yIo namespace to read the file “zdata.txt,” presented in figure 1, into a character array. The resulting character array is then used to create a ZDATA struct using the StringToZdata() function. Next, the newly created ZDATA struct is used to create a ZSTATS struct using the AnalyzeZdata() function. Finally, the contents of the ZSTATS struct are printed to the screen.

```
#include "y_zdata.h"//.............................yZdata,<cstdio>{printf()}
#include "y_io.h"//.....................................................yIo
int main(){
  char*s=yIo::ReadTextFile("zdata.txt");
  yZdata::ZDATA Z=yZdata::StringToZdata(s);
  yZdata::ZSTATS S=yZdata::AnalyzeZdata(Z);
  printf("M=%f grains\nN=%f\nKE=%f grains*feet^2/second^2\nm_avg=%f grains\n"
    "v_avg=%f feet/second\n",S.M,S.N,S.KE,S.m_avg,S.v_avg);
  delete[]s;
}//~~~~~~YAGENAUT@GMAIL.COM~~~~~~~~~~~~~~~~~~~~~~~~~LAST~UPDATED~16SEP2013~~~~~~
```

OUTPUT:

```
M=29738.329240 grains
N=581.116400
KE=142875248760.490690 grains*feet^2/second^2
m_avg=51.174479 grains
v_avg=3099.810999 feet/second
```

8. Summary

A summary sheet is provided as an appendix in this report. It presents the yZdata namespace, which contains the two structs and three functions that are described in this report. Also included is an example that incorporates the three examples presented in this report.

Appendix. yZdata Summary

yZdata Summary

y_zdata.h

```cpp
#ifndef Y_ZDATA_GUARD//        See Yager, R. J. "Reading, Writing, and Anaylzing
#define Y_ZDATA_GUARD//                                     ZDATA Files Using C++"
#include <cmath>//.......................................................sqrt()
#include <cstdio>//...................................................sprintf()
#include <cstdlib>//...............................................atoi(),atof()
#include <cstring>//..................................................strtok()
#include <vector>//.......................................................vector
namespace yZdata{//@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
  struct ZDATA{//<=====STORES THE INFORMATION CONTAINED IN A STANDARD ZDATA FILE
    std::vector<double>AL;//<------------------------LOWER ANGLES [ANGLE INDEX]
    std::vector<double>AM;//<---------------------MIDPOINT ANGLES [ANGLE INDEX]
    std::vector<double>AU;//<------------------------UPPER ANGLES [ANGLE INDEX]
    std::vector<double>VL;//<--------------SPEEDS FOR LOWER ANGLES [ANGLE INDEX]
    std::vector<double>VM;//<-----------SPEEDS FOR MIDPOINT ANGLES [ANGLE INDEX]
    std::vector<double>VU;//<--------------SPEEDS FOR UPPER ANGLES [ANGLE INDEX]
    std::vector<std::vector<double> >M;//--MASS VALUES [ANGLE INDEX][MASS INDEX]
    std::vector<std::vector<double> >N;//<------COUNTS [ANGLE INDEX][MASS INDEX]
    double sf;//<------------------------SHAPE FACTOR (USED FOR COMPUTING DRAG)
  };//~~~YAGENAUT@GMAIL.COM~~~~~~~~~~~~~~~~~~~~~~~~~LAST~UPDATED~16SEP2013~~~~~~
  struct ZSTATS{//<=============A SET OF VARIABLES THAT SUMMARIZES A ZDATA FILE
    double M;//<----------------------------------------TOTAL MASS OF FRAGMENTS
    double N;//<--------------------------------------TOTAL NUMBER OF FRAGMENTS
    double KE;//<-----------------------------------------TOTAL FRAGMENT ENERGY
    double m_avg;//<------------------------------------AVERAGE MASS OF FRAGMENTS
    double v_avg;//<-------------------------------------AVERAGE FRAGMENT SPEED
  };//~~~YAGENAUT@GMAIL.COM~~~~~~~~~~~~~~~~~~~~~~~~~LAST~UPDATED~16SEP2013~~~~~~
  inline char*ZdataToString(//<=========CREATES A CHAR ARRAY FROM A ZDATA STRUCT
      const ZDATA&Z,//<-------------------------------------------A ZDATA STRUCT
      int d=3){//<-THE NUMBER OF DIGITS AFTER THE DECIMAL POINT FOR NON INTEGERS
    int m=Z.M.size(),l=32;/*<-*/for(int i=0;i<m;++i)l+=73+142*Z.M[i].size()/7;
    char*s=new char[l+1];
    s+=sprintf(s,"ZDATA     %10d\n",m);//...........................header line
    for(int i=0,n=Z.M[0].size();i<m;++i,n=Z.M[i].size()){
      s+=sprintf(s," %9.*f %9.*f %9.*f %9.*f %9.*f %9.*f%10d\n",d,Z.AL[i],d,
        Z.AM[i],d,Z.AU[i],d,Z.VL[i],d,Z.VM[i],d,Z.VU[i],Z.M[i].size());
      for(int j=0;j<n;++j)
        s+=sprintf(s," %9.*f%s",d,Z.M[i][j],j%7==6||j==n-1?"\n":"");//....masses
      for(int j=0;j<n;++j)
        s+=sprintf(s," %9.*f%s",d,Z.N[i][j],j%7==6||j==n-1?"\n":"");}//...counts
    s+=sprintf(s," %9.*f\n",d,Z.sf);//..............................shape factor
    return s-l;//..................note that s points to newly allocated memory
  }//~~~~YAGENAUT@GMAIL.COM~~~~~~~~~~~~~~~~~~~~~~~~~LAST~UPDATED~16SEP2013~~~~~~
  inline ZDATA StringToZdata(//<======CREATES A ZDATA STRUCT FROM A ZDATA STRING
      char*s){//<-------------------A POINTER TO A STRING READ FROM A ZDATA FILE
    char d[6]="\n\r\f \t";//...........delimiting characters used in ZDATA file
    strtok(s,d);
    int m=atoi(strtok(NULL,d));//........................number of angular bins
    ZDATA Z={std::vector<double>(m),std::vector<double>(m),
      std::vector<double>(m),std::vector<double>(m),std::vector<double>(m),
      std::vector<double>(m),std::vector<std::vector<double> >(m),
      std::vector<std::vector<double> >(m)};
    for(int i=0;i<m;++i){
      Z.AL[i]=atof(strtok(NULL,d)),Z.AM[i]=atof(strtok(NULL,d)),
      Z.AU[i]=atof(strtok(NULL,d)),Z.VL[i]=atof(strtok(NULL,d)),
      Z.VM[i]=atof(strtok(NULL,d)),Z.VU[i]=atof(strtok(NULL,d));
      int n=atoi(strtok(NULL,d));//...number of mass bins in the jth angular bin
      for(int j=0;j<n;++j)Z.M[i].push_back(atof(strtok(NULL,d)));//.......masses
      for(int j=0;j<n;++j)Z.N[i].push_back(atof(strtok(NULL,d)));}//......counts
    Z.sf=atof(strtok(NULL,d));//..................................shape factor
    return Z;
  }//~~~~YAGENAUT@GMAIL.COM~~~~~~~~~~~~~~~~~~~~~~~~~LAST~UPDATED~16SEP2013~~~~~~
  inline ZSTATS AnalyzeZdata(//<======CREATE A ZSTATS STRUCT FROM A ZDATA STRUCT
      const ZDATA&Z){//<----------------------------CREATE USING ReadZdataFile()
    ZSTATS S={0,0,0,0,0};//....................................initialize values
    for(int i=0,m=Z.M.size();i<m;++i){//..............loop through angular zones
      double v_avg_2=(Z.VL[i]*Z.VL[i]+Z.VM[i]*Z.VM[i]+Z.VU[i]*Z.VU[i])/3;
      for(int j=0,n=Z.M[i].size();j<n;++j){//.............loop through mass bins
        S.N+=Z.N[i][j];//.....................................total fragment count
        S.M+=Z.N[i][j]*Z.M[i][j];//..........................total fragment mass
        S.KE+=.5*Z.M[i][j]*Z.N[i][j]*v_avg_2;}}//...........total kinetic energy
    S.m_avg=S.M/S.N;//....................................average fragment mass
    S.v_avg=sqrt(2*S.KE/S.M);//...........................average fragment speed
    return S;
  }//~~~~YAGENAUT@GMAIL.COM~~~~~~~~~~~~~~~~~~~~~~~~~LAST~UPDATED~16SEP2013~~~~~~
}//@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
#endif
```

Example

```cpp
#include <cstdlib>//...........................................rand(),RAND_MAX
#include "y_zdata.h"//..........................................yZdata,<vector>
#include "y_io.h"//.........................................................yIo
int main(){
  using std::vector;
  int m=12,n=10;//number of angular bins and number of mass bins per angular bin
  yZdata::ZDATA Z={vector<double>(m),vector<double>(m),vector<double>(m),//setup
    vector<double>(m),vector<double>(m),vector<double>(m),//                   a
    vector<vector<double> >(m,vector<double>(n)),//                        ZDATA
    vector<vector<double> >(m,vector<double>(n)),.5};//                   struct
  for(int i=0;i<m;++i){//..............................populate the ZDATA struct
    Z.AL[i]=i*180./m,Z.AM[i]=(i+.5)*180./m,Z.AU[i]=(i+1)*180./m;
    Z.VL[i]=3000+rand()*200./RAND_MAX;//..............set upper, middle, & lower
    Z.VM[i]=3000+rand()*200./RAND_MAX;//                speeds to random values
    Z.VU[i]=3000+rand()*200./RAND_MAX;//                  between 3000 and 3200
    for(int j=0;j<n;++j){
      Z.M[i][j]=rand()*100./RAND_MAX;//.....set masses to random values 0 to 100
      Z.N[i][j]=rand()*10./RAND_MAX;}}//.....set counts to random values 0 to 10
  char*s=yZdata::ZdataToString(Z,3);
  yIo::WriteTextFile("zdata.txt",s);
  delete[]s;
  char*s2=yIo::ReadTextFile("zdata.txt");
  yIo::WriteTextFile("zdata2.txt",s);
  yZdata::ZDATA Z2=yZdata::StringToZdata(s2);
  yZdata::ZSTATS S=yZdata::AnalyzeZdata(Z2);
  printf("M=%f grains\nN=%f\nKE=%f grains*feet^2/second^s\nm_avg=%f grains\n"
    "v_avg=%f feet/second\n",S.M,S.N,S.KE,S.m_avg,S.v_avg);
  delete[]s2;
  return 0;
}//~~~~~~YAGENAUT@GMAIL.COM~~~~~~~~~~~~~~~~~~~~~~~~~LAST~UPDATED~16SEP2013~~~~~~
```

Example Output

```
M=29738.329240 grains
N=581.116400
KE=142875248760.490690 grains*feet^2/second^2
m_avg=51.174479 grains
v_avg=3099.810999 feet/second
```

NO. OF
COPIES ORGANIZATION

1 (PDF) DEFENSE TECHNICAL INFORMATION CTR DTIC OCA

1 (PDF) DIRECTOR US ARMY RESEARCH LAB IMAL HRA

1 (PDF) DIRECTOR US ARMY RESEARCH LAB RDRL CIO LL

1 (PDF) GOVT PRINTG OFC A MALHOTRA

ABERDEEN PROVING GROUND

1 (PDF) DIR USARL RDRL WML A R YAGER

INTENTIONALLY LEFT BLANK.